Discrete Applied Mathematics 17 (1987) 187–211
North-Holland

A DECOMPOSITION THEORY BASED ON A DOMINANCE RELATION AND COMPOSITE JOBS

Yasuki SEKIGUCHI
Faculty of Economics, Hokkaido University, Kita-ku Kita 9 Nishi 7, Sapporo 060, Japan

Received February 1985
Revised November 1985

A decomposition theory which includes Sidney's decomposition theory as a special case is constructed by using new fundamentals such as a dominance relation which solves a sequencing problem without precedence constraints, a unique interpretation rule which restricts application of the dominance relation and a composite job (for a sequence of jobs) which plays a central role in developing a new decomposition algorithm. It applies to all problems involved in Sidney's theory. Moreover, it involves a problem which Sidney's theory does not involve, i.e., a class of problems equivalent to a one-machine minimum total weighted-completion time problem. The theory shows also that a variety of tie-breaking rules can be used in the decomposition algorithm, though it was uniquely determined in the previous theory.

1. Introduction

Studies about sequencing problems under precedence constraints had conventionally developed many individual theories for individual problems. The theory of a series-parallel algorithm by Monma and Sidney [8] and the theory of a decomposition algorithm by Sidney [15], which have their origin in individual theories such as those in [14], [9] and [13], are valuable in order to overcome precedence constraints. However, these theories were constructed in a form which can apply to various problems which individual theories had been developed for. Thus, they should be rated high as general (or unifying) theories.

This paper is devoted to show the usefulness of a concept of composite jobs in order to further develop these general theories.

A composite job, which is a suppositional job defined with respect to a string (i.e., a sequence of jobs), had been first introduced by Sidney [13]. Then, Kurisu [4], Lawler [5], Sidney [14] and Maggu et al. [6] had successfully utilized this concept with or without modifications, in order to reduce a complicated problem into an equivalent and simpler problem.

In [10] and [11], the author proposed a new definition of composite jobs, and used it for inventing a unified expression and a general solution to various two-machine sequencing problems minimizing makespan. A similar approach will simplify Sidney's theory, which uses strings as fundamental objectives of analysis. Our theory is simple in a sense that a dominance relation (i.e., a binary relation)

is defined only on the set of jobs, whereas it is defined on the set of strings as well as on the set of jobs in Sidney's theory.

Sidney [15] realized his generalization by introducing an interval order for an original binary relation. It is understood that the interval order was used for giving implicit transitivity to the original binary relation which is complete but not transitive. However, his success was partial in a sense that his general theory does not apply to the one-machine minimum total weighted-completion time problem for which he had first invented a decomposition algorithm [13]. Instead of the interval order, we use a new idea of a unique interpretation rule (UIR). This makes it possible to use a variety of tie-breaking rules in a decomposition algorith, though it was uniquely determined in Sidney's theory. Besides, our theory includes a class of problems equivalent to the one-machine minimum total weighted-completion time problem.

In the next section, notations are introduced and sequencing problems with and without precedence constraints are defined. In Section 3, a dominance relation which solves a sequencing problem without precedence constraints is introduced. Its definition and a restriction on its application constitute a part of the core of the proposed theory. The reminder of the core consists of composite jobs and their properties, which are also introduced in the section. In Section 4 is clarified a sufficient condition under which a sequencing problem can be solved by a decomposition algorithm. Various problems in Section 5 illustrate applicability of the proposed theory. The final section gives a summary.

2. Basic definitions and notation

S is the set of all elements with a set of specific attributes. N is a subset of S. Suppose that elements in N are numbered. For the sake of simplicity, N is supposed to represent the set of the numbers, also. Elements i and j of S are said to be *equivalent* to each other, denoted as $i \equiv j$, if their values of the attributes are equal to each other. S and N can include an arbitrary number of equivalent elements.

An element in S is called a *job* and a permutation of jobs in a subset of S is called a (*sub-*)*schedule* or a *string*. The i-th job in a schedule is designated by $\langle i \rangle$. The set of jobs included in schedule a is represented by $\{a\}$. The symbol $a(K)$ implies that a is a schedule of jobs in K, i.e., $K = \{a\}$.

The schedule where schedule a is directly followed by another b is called the concatenation of a and b, and is designated by $a \cdot b$. Whenever the concatenation of a and b is referred to, each job in $\{a\} \cap \{b\}$ is assumed to be two equivalent and different jobs. The length of a is the number of jobs in $\{a\}$.

A cost function g for evaluating schedules is a mapping from an arbitrary schedule to a real value. A place to process jobs is called a *shop*. It is natural to suppose that a shop has a *state* which varies as jobs are processed. A value of g for a is $g(a : \mathrm{IS})$, where IS is the state of the shop at the beginning of the process of a.

It is also natural to think that a *standard initial state* IS_0 can be determined for every shop. The value of g for a under the standard initial state is denoted as $f(a)$, i.e., $f(a)=g(a:IS_0)$. The function f is called a *sequencing function*, and hereafter f is used instead of g. Only when

$$f(a)\leq f(b) \quad \text{implies} \quad g(a:IS)\leq g(b:IS) \quad \text{for all IS}$$

holds for every $\{a\}=\{b\}\subset S$, f can be used as a substitute for g. It is a usual practice to use f instead of g in mathematical formulations of sequencing problems. However, some problems treated in this manner do not satisfy this condition, e.g., the three-machine minimum makespan flow-shop problem.

A sequencing function f is said to satisfy the *series network decomposition* (SND) property if for every $\{b\}=\{b'\}\subset S$,

$$f(b)\leq f(b') \quad \text{implies} \quad f(a\cdot b\cdot c)\leq f(a\cdot b'\cdot c) \quad \text{for all } \{a\},\{c\}\subset S.$$

Let $Q(N)$ be the set of all schedules of jobs in N. A *free problem* $P(N)$ is to find a schedule in $Q(N)$ which minimizes a sequencing function:

$$P(N) \qquad \underset{a\in Q(N)}{\text{minimize}}\, f(a).$$

An acyclic network, $G=(N,E)$, with a node set N and an arc set $E\subset N\times N$ is called a *precedence network*. $(i,j)\in E$ implies that i must be processed prior to j. $Q(N,G)$ denotes the set of schedules which are in $Q(N)$ and satisfy the precedence constraint designated by G. A constrained problem is

$$P(N,G) \qquad \underset{a\in Q(N,G)}{\text{minimize}}\, f(a).$$

$G_K=(K,E)$ represents a sub-network of G induced by $K\subset N$. For the sake of simplicity, $P(K,G)$ denotes $P(K,G_K)$, which is a sub-problem of $P(N,G)$.

$K\subset N$ is called an *initial set of* G if no job in $N\setminus K$ precedes jobs in K, and is called a *terminal set of* G if no job in $N\setminus K$ succeeds jobs in K. An arc (i,j) of G is a chain if both out-degree of i and the in-degree of j are 1.

A binary relation $\boldsymbol{D}$ is said to be *complete* if $i\boldsymbol{D}j$ and/or $j\boldsymbol{D}i$ hold for all $\{i,j\}\subset S$. $\boldsymbol{D}$ is said to be *transitive* if $i\boldsymbol{D}j$ and $j\boldsymbol{D}k$ imply $i\boldsymbol{D}k$ for all $\{i,j,k\}\subset S$. Let $i\boldsymbol{D}j$ denote a case where $i\boldsymbol{D}j$ holds but $j\boldsymbol{D}i$ does not. Then, $\boldsymbol{D}$ is said to be *weakly transitive* if

$$i\boldsymbol{D}j \text{ and } j\boldsymbol{D}k \text{ imply } i\boldsymbol{D}k,$$

$$i\boldsymbol{D}j \text{ and } j\boldsymbol{D}k \text{ imply } i\boldsymbol{D}k, \text{ and}$$

$$i\boldsymbol{D}j \text{ and } j\boldsymbol{D}k \text{ imply } i\boldsymbol{D}k$$

hold for every $\{i,j,k\}\subset S$. The difference between the weak transitivity and the transitivity is that the former does not guarantee $i\boldsymbol{D}k$ even if $i\boldsymbol{D}j$ and $j\boldsymbol{D}k$ hold. If $\boldsymbol{D}$ is transitive, the following simplification of a notation is used:

$$i\boldsymbol{D}j\boldsymbol{D}k \quad \text{for} \quad i\boldsymbol{D}j \text{ and } j\boldsymbol{D}k.$$

$\boldsymbol{D}$ is said to be reasonable if for every $\{i,j\}\subset S$,

$$iDj \quad \text{iff} \quad f(a\cdot i\cdot j\cdot b)\leq f(a\cdot j\cdot i\cdot b) \quad \text{for all } \{a\},\{b\}\subset S.$$

$\boldsymbol{D}$ is said to be *weakly reasonable* if for every $\{i,j\}\subset S$,

$$iDj \quad \text{implies} \quad f(a\cdot i\cdot j\cdot b)\leq f(a\cdot j\cdot i\cdot b) \quad \text{for all } \{a\},\{b\}\subset S.$$

The *composite job* for schedule b is $J_b\in S$ such that

$$f(a\cdot b\cdot c)=f(a\cdot J_b\cdot c)+\Delta_b \quad \text{for all } \{a\},\{c\}\subset S,$$

where Δ_b is a constant which does not depend on a and c. A procedure to determine a composite job for a schedule is called a *composition* (of jobs in the schedule).

A reverse problem (McMahon and Burton [7], or a dual problem [15]) of $P(N,G)$ is defined below.

$\tilde{G}$ denotes the network obtained from $G=(N,E)$ by reversing the directions of all arcs of G, i.e., $\tilde{G}=(N,\tilde{E})$ where $\tilde{E}=\{(j,i):(i,j)\in E\}$. Let $\tilde{a}$ be the schedule obtained by reversing the order of jobs in a, and define the reverse sequencing function $\tilde{f}$ as $\tilde{f}(\tilde{a})=f(a)$ for any schedule a. Note that $\tilde{a}\in Q(N,\tilde{G})$ if and only if $a\in Q(N,G)$. Naturally,

$$\tilde{f}(\tilde{a})\leq\tilde{f}(\tilde{b}) \quad \text{iff} \quad f(a)\leq f(b) \quad \text{for } \tilde{a},\tilde{b}\in Q(N,\tilde{G}).$$

The *reverse problem* $\tilde{P}(N,\tilde{G})$ of the *original problem* $P(N,G)$ is defined by

$$\tilde{P}(N,\tilde{G}) \qquad \underset{\tilde{a}\in Q(N,\tilde{G})}{\text{minimize}}\ \tilde{f}(\tilde{a}).$$

Moreover, define the composite job $J_{\tilde{a}}$ for schedule $\tilde{a}$ as

$$J_{\tilde{a}}\equiv J_a \quad \text{and} \quad \Delta_{\tilde{a}}=\Delta_a.$$

We will use the two-machine minimum makespan flow-shop problem defined below as an illustrative example. Define the job set and the sequencing function as follows.

$$S=\{(A_i,B_i):A_i,B_i \text{ are non-negative integers}\},$$

$$T_1(a\cdot i)=T_1(a)+A_i, \qquad T_2(a\cdot i)=\max\{T_1(a\cdot i),T_2(a)\}+B_i,$$

$$f(a)=T_2(a),$$

where $T_1(\emptyset)=T_2(\emptyset)=0$. Given a finite subset $N\subset S$ and a precedence constraint G, $P(N,G)$ is the problem to find a schedule minimizing f in $Q(N,Q)$.

The composite job $J_a=(A_a,B_a)$ for schedule a is defined by

$$A_a=f(a)-\sum_{i\in\{a\}}B_i, \qquad B_a=f(a)-\sum_{i\in\{a\}}A_i \quad \text{and} \quad \Delta_a=\sum_{i\in\{a\}}(A_i+B_i)-f(a).$$

This definition has been shown to satisfy the condition of a composite job for a [10].

3. A dominance relation and composite jobs

A sequencing function is said to satisfy the *weak adjacent pairwise interchange* (WAPI) property if there is a complete, transitive and weakly reasonable binary relation defined on S. If a sequencing function satisfies the WAPI property, $P(N)$ is solved by Smith's sequencing theorem [16], and an optimal solution can be made by a sorting algorithm. Because of this efficiency, a binary relation with the three properties is usually searched for when an unsolved $P(N)$ is given. It is a popular and effective strategy to derive a sufficient condition for

$$f(a\cdot i\cdot j\cdot b)\leq f(a\cdot j\cdot i\cdot b) \quad \text{for all } \{a\},\{b\}\subset S.$$

Actually, this try often results in obtaining a *dominance relation* $\boldsymbol{R}$ which is a binary relation with the following properties:

I. $\boldsymbol{R}$ is complete, weakly transitive and reasonable.

II. $S=S_1\cup S_2$ and for $i\in S_1\setminus S_2$, $j\in S_1\cap S_2$ and $k\in S_2\setminus S_1$, $i\boldsymbol{R}j$, $j\boldsymbol{R}k$ and $i\boldsymbol{R}k$.

III. Two binary relations $\boldsymbol{R}_u$ $(u=1,2)$ with the following properties can be defined on S_u.

III.1. $\boldsymbol{R}_u$ is complete, transitive and weakly reasonable.

III.2. If $i,j\in S_1\cap S_2$, $i\boldsymbol{R}_1 j$ if and only if $j\boldsymbol{R}_2 i$.

III.3. If $i\boldsymbol{R}j$ and $jR_1 i$, then $j\in S_1\cap S_2$. If $i\boldsymbol{R}j$ and $jR_2 i$, then $i\in S_1\cap S_2$.

Theorem 1. (1) *$i,j\in S_1\cap S_2$ implies $i\boldsymbol{R}j$ and $j\boldsymbol{R}i$.*

(2) *iRj implies $i\in S_1\setminus S_2$ and $j\in S_2\setminus S_1$, or*
$i\in S_1\setminus S_2$, $j\in S_1$ and $iR_1 j$, or
$i\in S_2$, $j\in S_2\setminus S_1$ and $iR_2 j$.

(3) *Let f satisfy the SND property, then iRj implies $f(i\cdot j)<f(j\cdot i)$.*

(4) *Let f satisfy the SND property, then*
$i\in S_1\setminus S_2$ and $iR_1 j$ imply $f(i\cdot j)<f(j\cdot i)$, and
$j\in S_2\setminus S_1$ and $iR_2 j$ imply $f(i\cdot j)<f(j\cdot i)$.

Proof. (1) $\boldsymbol{R}$ is reasonable and $\boldsymbol{R}_u$ is weakly reasonable. Therefore, from III.2, immediate.

(2) If $i,j\in S_1\cap S_2$, iRj never happens, because of (1). Therefore, if iRj, i and/or j must be in $S_1\setminus S_2$ or $S_2\setminus S_1$. If $i\in S_1\setminus S_2$, either $j\in S_1$ or $j\in S_2\setminus S_1$. If the latter, then the first case of the theorem fits, and if the former, then the second case. If $i\in S_1\cap S_2$ or $i\in S_2\setminus S_1$, then $j\in S_2\setminus S_1$; the third case.

(3) Assume the contrary. From the reasonability of $\boldsymbol{R}$, $f(i\cdot j)=f(j\cdot i)$. Then, by the SND property, $f(a\cdot i\cdot j\cdot b)=f(a\cdot j\cdot i\cdot b)$ for all $\{a\},\{b\}\subset S$; a contradiction to the reasonability of $\boldsymbol{R}$.

(4) Both i and j are in S_1 if $iR_1 j$. A contradiction to III.3 if both $j\boldsymbol{R}i$ and $iR_1 j$ for $i\in S_1\setminus S_2$. Therefore, iRj. By (3), the proof has been completed for the first case. Similarly, the second case is done. □

Theorem 2. *Assume that iRj, jRk and kRi. Then, $j\in S_1\cap S_2$. Moreover, one of the*

following three cases occurs: iR_2j, jR_1k and at most two of i, j and k are in $S_1 \cap S_2$; jR_1i, jR_1k and $i,k \in S_1 \setminus S_2$; iR_2j, kR_2j and $i,k \in S_2 \setminus S_1$.

Proof. Assume iR_uj and jR_vk $(u,v=1,2)$. If iR_uj and jR_uk, then iR_uk by the transitivity. Therefore, let $u \neq v$. Then, $i,j \in S_u$ and $j,k \in S_v$; this implies $j \in S_1 \cap S_2$. If all three are in $S_1 \cap S_2$, then kRi contradicts (1) of Theorem 1. This completes the proof of the first part of the theorem. The remaining part can be proved similarly. □

A generalized sequencing theorem [15] proved for an interval order can be proved for the dominance relation, also. If f has the dominance relation, f is said to satisfy the *adjacent pairwise interchange* (API) property.

Theorem 3 (A generalized sequencing theorem [15, Theorem 2].) *Let f satisfy the API property. A schedule $a \in Q(N)$ such that i precedes j if iRj is an optimal solution of $P(N)$.*

Proof. The proof can be similar to Sidney [15]; however, one that uses the equivalency of $\boldsymbol{R}$ to an interval order is given.

Suppose that an interval $[r_{\min}(i), r_{\max}(i)]$ such that

$$r_{\min}(i) = r_{\max}(i) \in Z_- \ (Z_- = \{x: x<0\} \cup \{-\infty\}) \quad \text{if } i \in S_1 \setminus S_2.$$

$$r_{\min}(i) = r_{\max}(i) \in Z_+ \ (Z_+ = \{x: x \geq 0\} \cup \{+\infty\}) \quad \text{if } i \in S_2 \setminus S_1,$$

and

$$r_{\min}(i) \in Z_-, \quad r_{\max}(i) \in Z_+ \quad \text{if } i \in S_1 \cap S_2,$$

is associated with each $i \in S$. Then, an interval order $\boldsymbol{R}_I$ defined by

$$i\boldsymbol{R}_Ij \quad \text{iff} \quad r_{\min}(i) \leq r_{\max}(j)$$

is a Sidney's interval order. If f has a dominance relation, by virtue of the definition of $\boldsymbol{R}$, it is possible to define an interval for each $i \in S$ such that

$$r_{\min}(i) \leq r_{\min}(j) \quad \text{for } i,j \in S_1 \quad \text{iff} \quad i\boldsymbol{R}_1j,$$

$$r_{\max}(i) \leq r_{\max}(j) \quad \text{for } i,j \in S_2 \quad \text{iff} \quad i\boldsymbol{R}_2j.$$

Then, for $\boldsymbol{R}_I$ determined by the intervals, $i\boldsymbol{R}_Ij$ if and only if $i\boldsymbol{R}j$. Thus, $\boldsymbol{R}$ is shown to be an interval order. Contrary to this, an interval order may not satisfy the properties II and III even when f satisfies the API property in the exact sense of Sidney [15]. In the proof of Sidney, the fact is used that an interval order defined by

$$i\boldsymbol{R}_I^*j \quad \text{iff} \quad r_{\min}(i) \leq r_{\min}(j)$$

is complete, transitive, weakly reasonable and consistent with $\boldsymbol{R}_I$, i.e.,

$$i\boldsymbol{R}_I^*j \quad \text{implies} \quad i\boldsymbol{R}_Ij. \qquad \square$$

As a solution of $P(N)$, a fatal defect of $\boldsymbol{R}$ is the non-transitivity. Assume that i, j and k are those in Theorem 2, and that pair i, j is first tested with result $i\boldsymbol{R}j$ and second pair j, k with result $j\boldsymbol{R}k$. (However, notice that the results could be $j\boldsymbol{R}i$ and/or $k\boldsymbol{R}j$ according to Theorem 2.) Then, many sorting algorithms give a non-optimal sequence $i \cdot j \cdot k$. An essential factor for this inconvenience to happen is a tie-breaking rule concerning $j \in S_1 \cap S_2$. No problem emerges if either $j\boldsymbol{R}i$ or $k\boldsymbol{R}j$ is selected.

A simple tie-breaking rule which avoids this problem is to select $i\boldsymbol{R}j$ whenever $i \in S_1 \cap S_2$ is compared with $j \in S_2 \setminus S_1$ and to use $\boldsymbol{R}_1$ for $i, j \in S_1$. This is equivalent to using $\boldsymbol{R}_I^*$ instead of $\boldsymbol{R}$. Another rule is to select $i\boldsymbol{R}j$ whenever $j \in S_1 \cap S_2$ is compared with $i \in S_1 \setminus S_2$ and to use $\boldsymbol{R}_2$ for $i, j \in S_2$. These are special cases of the following general technique.

Determine C_1 and C_2 arbitrarily such that

$$C_1 \cup C_2 = S_1 \cap S_2 \quad \text{and} \quad C_1 \cap C_2 = \emptyset.$$

Define a binary relation $\boldsymbol{R}^*$ as

$$i\boldsymbol{R}^*j \quad \text{if } i\boldsymbol{R}_1 j \text{ and } \{i, j\} \subset S_1^*, \text{ or } i\boldsymbol{R}_2 j \text{ and } \{i, j\} \subset S_2^*,$$

$$i\boldsymbol{R}^*j \quad \text{for } i \in S_1^* \text{ and } j \in S_2^*,$$

where,

$$S_1^* = (S_1 \setminus S_2) \cup C_1 \quad \text{and} \quad S_2^* = (S_2 \setminus S_1) \cup C_2.$$

Let $C_1 = \emptyset$ ($C_2 = \emptyset$). Then the latter (former) of the illustrated tie-breaking rules emerges. Anyhow, $\boldsymbol{R}^*$ can be induced by restricting $\boldsymbol{R}$.

Theorem 4. *If f has the API property, then $\boldsymbol{R}^*$ is complete, transitive and weakly reasonable.*

Proof. (Completeness) Evident.

(Transitivity) Assume that $i\boldsymbol{R}^*j$ and $j\boldsymbol{R}^*k$. If $k \in S_1^*$, then $i, j \in S_1^*$ by the definition of $\boldsymbol{R}^*$, and the transitivity of $\boldsymbol{R}_1$ guarantees the transitivity of $\boldsymbol{R}^*$. Let $k \in S_2^*$. If $i \in S_2^*$, $\boldsymbol{R}^*$ is proved to be transitive by a similar reasoning. If $i \in S_1^*$, then $i\boldsymbol{R}^*k$ is evident.

(Weak reasonability) Evident, because $i\boldsymbol{R}^*j$ only if $i\boldsymbol{R}j$. □

There are three cases where $\boldsymbol{R}^*$ is different from $\boldsymbol{R}$:

Case 1. If $i \in S_1 \setminus S_2$ and $j \in C_2$, then $i\boldsymbol{R}^*j$. However, both $i\boldsymbol{R}j$ and $j\boldsymbol{R}i$ may hold.

Case 2. If $i \in C_1$ and $j \in C_2$, then $i\boldsymbol{R}^*j$. However, both $i\boldsymbol{R}j$ and $j\boldsymbol{R}i$ always hold.

Case 3. If $i \in C_1$ and $j \in S_2 \setminus S_1$, then $i\boldsymbol{R}^*j$. However, both $i\boldsymbol{R}j$ and $j\boldsymbol{R}i$ may hold.

Theorem 2 tells that if both $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are applied for $j \in S_1 \cap S_2$, the transitivity may be violated. Taking into account these facts, a practical technique to derive $\boldsymbol{R}^*$ from $\boldsymbol{R}$ while $P(N)$ is being solved can be the following. Remember that no tie case occurs if $i \in S_1 \setminus S_2$ and $j \in S_2 \setminus S_1$.

A practical tie-breaking rule

(Use $\boldsymbol{R}_1$ or $\boldsymbol{R}_2$ when two jobs with no flag are compared with each other.)

(1) When $i \in S_1 \cap S_2$ with no flag is compared with $j \in S_1$ and the result is $i\boldsymbol{R}j$ according to $i\boldsymbol{R}_1 j$, then put flag 1 on i.

(2) When $i \in S_1 \cap S_2$ with no flag is compared with $j \in S_2$ and the result is $j\boldsymbol{R}i$ according to $j\boldsymbol{R}_2 i$, then put flag 2 on i.

(3) If i has flag 1, then use $\boldsymbol{R}_1$ for comparing it with $j \in S_1$ and select $i\boldsymbol{R}j$ for $j \in S_2 \setminus S_1$.

(4) If i has flag 2, then use $\boldsymbol{R}_2$ for comparing it with $j \in S_2$ and select $j\boldsymbol{R}i$ for $j \in S_1 \setminus S_2$.

(5) If i and j have flag 1 and 2, respectively, then select $i\boldsymbol{R}j$.

The above technique is equivalent to using $\boldsymbol{R}^*$ for $\boldsymbol{R}$ in the following sense. Assume that $P(N)$ has been solved using $\boldsymbol{R}$ under the tie-breaking rule. Let N_1 (N_2) be a set of jobs with flag 1 (2). Determine C_1 (C_2) such that $N_1 \subset C_1$ and $N_2 \subset C_2$. The solution process with $\boldsymbol{R}$ can be interpreted as one with $\boldsymbol{R}^*$ defined by these C_1 and C_2. Notice that (1) and (3) ((2) and (4)) in the tie-breaking rule prohibits $\boldsymbol{R}_2$ ($\boldsymbol{R}_1$) from being used for $i \in S_1 \cap S_2$ for which $\boldsymbol{R}_1$ ($\boldsymbol{R}_2$) has already been used.

In this context, a rule determining C_1 and C_2 is called a *unique interpretation rule* (UIR), and $\boldsymbol{R}$ is said to become $\boldsymbol{R}^*$ under a UIR.

For the flow-shop problem, let $S_1 = \{(A_i, B_i) \in S : A_i \leq B_i\}$ and $S_2 = \{(A_i, B_i) \in S : A_i \geq B_i\}$. Its dominance relation is:

$$
\begin{aligned}
i\boldsymbol{R}j \quad \text{iff} \quad & i, j \in S_1 \text{ and } A_i \leq A_j, \text{ or} && (\boldsymbol{R}_1)\\
& i \in S_1 \text{ and } j \in S_2, \text{ or} \\
& i, j \in S_2 \text{ and } B_i \geq B_j. && (\boldsymbol{R}_2)
\end{aligned}
$$

$\boldsymbol{R}^*$ is defined as follows:

$$
\begin{aligned}
i\boldsymbol{R}^*j \quad \text{iff} \quad & i, j \in S_1^* \text{ and } A_i \leq A_j, \text{ or} \\
& i \in S_1^* \text{ and } j \in S_2^*, \text{ or} \\
& i, j \in S_2^* \text{ and } B_i \geq B_j.
\end{aligned}
$$

Let $(A_i, B_i) = (5, 5)$, $(A_j, B_j) = (2, 2)$ and $(A_k, B_k) = (3, 4)$. Then, $i\boldsymbol{R}j$ by the third (or second) line, $j\boldsymbol{R}k$ by the first line, but $k\boldsymbol{R}i$ by the first (or second) line. Replacing (A_i, b_i) with $(3, 2)$ gives another example of transitivity violation. See that $\boldsymbol{R}^*$ recovers transitivity for these examples. $\boldsymbol{R}'$ below is also transitive, though it is not in the class of $\boldsymbol{R}^*$.

$$
\begin{aligned}
i\boldsymbol{R}'j \quad \text{iff} \quad & i, j \in S_1 \setminus S_2 \text{ and } A_i \leq A_j, \text{ or} \\
& i \in S_1 \setminus S_2 \text{ and } j \in S_2, \text{ or} \\
& i, j \in S_1 \cap S_2, \text{ or}
\end{aligned}
$$

$i \in S_1 \cap S_2$ and $j \in S_2 \setminus S_1$, or

$i, j \in S_2 \setminus S_1$ and $B_i \geq B_j$.

A composition is said to satisfy the *coincidency*, if

$$J_{12} \equiv J_{a1 \cdot a2} \quad \text{for all } \{a1\}, \{a2\} \subset S,$$

where J_{12} is the composite job for $J_{a1} \cdot J_{a2}$. The coincidency requires that the composite job for a schedule can be determined either by a method which exactly follows the definition, or by a method which at first divides the schedule into plural sub-schedules, then determines composite jobs for the sub-schedules and finally determines the composite job for the schedule of these composite jobs.

Theorem 5. *If a composition satisfies the coincidency, then*

$$\Delta_{12} + \Delta_{a1} + \Delta_{a2} = \Delta_{a1 \cdot a2}.$$

Proof. By the definition of composite jobs, for all $\{a\}, \{b\} \subset S$,

$$\begin{aligned} f(a \cdot a1 \cdot a2 \cdot b) &= f(a \cdot J_{a1} \cdot a2 \cdot b) + \Delta_{a1} = f(a \cdot J_{a1} \cdot J_{a2} \cdot b) + \Delta_{a1} + \Delta_{a2} \\ &= f(a \cdot J_{12} \cdot b) + \Delta_{12} + \Delta_{a1} + \Delta_{a2}. \end{aligned}$$

On the other hand,

$$f(a \cdot a1 \cdot a2 \cdot b) = f(a \cdot J_{a1 \cdot a2} \cdot b) + \Delta_{a1 \cdot a2}.$$

By the coincidency, $f(a \cdot J_{12} \cdot b) = f(a \cdot J_{a1 \cdot a2} \cdot b)$. Therefore, the theorem is simply derived. □

Let f satisfy the API property. A composition is said to satisfy the *invariant membership* (*INV*) *property* if for every $\{i, j\} \subset S$,

$$\begin{aligned} \{i, j\} \subset S_1 \text{ and } i \in S_1 \setminus S_2 \quad &\text{imply} \quad J_{i \cdot j}, J_{j \cdot i} \in S_1 \setminus S_2, \\ \{i, j\} \subset S_1 \cap S_2 \quad &\text{implies} \quad J_{i \cdot j}, J_{j \cdot i} \in S_1 \cap S_2, \\ \{i, j\} \subset S_2 \text{ and } i \in S_2 \setminus S_1 \quad &\text{imply} \quad J_{i \cdot j}, J_{j \cdot i} \in S_2 \setminus S_1. \end{aligned}$$

A composition is said to satisfy the $*$ *identical membership* ($*IDN$) *property* if for every $K \subset S$,

$$J_a \in S' \quad \text{for some } a \in Q(K) \quad \text{implies} \quad J_a \in S' \quad \text{for all } a \in Q(K),$$

where S' is one of $S_1 \setminus S_2$, $S_1 \cap S_2$ and $S_2 \setminus S_1$.

Put the following restriction on a UIR: if $J_a \in S_1 \cap S_2$ for some $a \in Q(K)$, then $J_a \in C_u$ for all $a \in Q(K)$.

(A decomposition algorithm discussed in the following section makes at most one composite job for $K \subset S$, and no practical problem emerges from this condition.) Then, the *IDN property gives composite jobs a convenient character:

(A) $J_a \in S_u^*$ for all $a \in Q(K)$ where $K \subset S$ and $u = 1$ or 2.

This simplifies a theoretical treatment below. Henceforth, a UIR is assumed to satisfy the above condition whenever necessary, i.e., the *IDN property actually implies (A).

A composition is said to satisfy the *consistent equivalency*, if for every $\{a\} = \{b\} \subset S$, $J_a \equiv J_b$. In other words, the consistent equivalency of a composition requires that a composite job for a schedule depends only on the jobs in the schedule but not on their sequence. The constant, Δ_a, may not be equal to Δ_b even if a composition satisfies the consistent equivalency. Notice that if a composition satisfies the consistent equivalency, it satisfies the *IDN property, too.

A composition is said to satisfy the *strict dominance conservativity* (SDC) if

$$iR_1 j \quad \text{implies} \quad J_{i\cdot j} R_1 j \text{ for } \{i,j\} \subset S_1, \quad \text{and}$$

$$iR_2 j \quad \text{implies} \quad iR_2 J_{i\cdot j} \text{ for } \{i,j\} \subset S_2.$$

A composition is said to satisfy the * *consistent monotonicity* under a UIR if the following hold for every $\{a\} = \{b\} \subset S$.

$$J_a R^* J_b \quad \text{iff} \quad f(a) < f(b), \quad \text{if } J_a, J_b \in S_1^*.$$

$$J_a R^* J_b \quad \text{iff} \quad f(a) > f(b), \quad \text{if } J_a, J_b \in S_2^*.$$

$$J_a \equiv J_b \quad \text{iff} \quad f(a) = f(b).$$

It is easy to give an example of the two-machine flow-shop problem which violates a consistent monotonicity which is defined by replacing $\boldsymbol{R}^*$ with $\boldsymbol{R}$ in the above definition. Notice that a composition can satisfy both the consistent equivalency and the * consistent monotonicity in only a trivial problem whose $f(a)$ is a constant for all $a \in Q(N)$.

A composition is said to satisfy the * *dominance conservativity* (*DC) if

$$i\boldsymbol{R}^* j \quad \text{implies} \quad i\boldsymbol{R}^* J_{i\cdot j} \boldsymbol{R}^* j \text{ and } i\boldsymbol{R}^* J_{j\cdot i} \boldsymbol{R}^* j.$$

Theorem 6. *Let f satisfy the API and SND properties. Then,*

$$i\boldsymbol{R}j \quad \textit{implies} \quad i\boldsymbol{R}J_{i\cdot j}, i\boldsymbol{R}J_{j\cdot i}, J_{i\cdot j}\boldsymbol{R}j \textit{ and } J_{j\cdot i}\boldsymbol{R}j.$$

Proof. Assume that $i\boldsymbol{R}j$ and $J_{i\cdot j}\boldsymbol{R}i$. By (3) of Theorem 1, $f(J_{i\cdot j}\cdot i) < f(i \cdot J_{i\cdot j})$. According to the definition of composite jobs,

$$f(J_{i\cdot j}\cdot i) = f(i\cdot j\cdot i) - \Delta_{i\cdot j} \quad \text{and} \quad f(i\cdot J_{i\cdot j}) = f(i\cdot i\cdot j) - \Delta_{i\cdot j}.$$

Combining the three equations, we get $f(i\cdot j\cdot i) < f(i\cdot i\cdot j)$. On the other hand, the reasonability of $\boldsymbol{R}$ implies $f(i\cdot i\cdot j) \leq f(i\cdot j\cdot i)$, a contradiction. Therefore, $i\boldsymbol{R}J_{i\cdot j}$ must hold. The remaining three relations can be proved similarly. □

Any composition does not satisfy the *DC if a UIR is inadequate, e.g., assume

that $i, j \in C_1$ and $J_{i \cdot j} \in C_2$. Even if a UIR is adequate, the *DC does not necessarily hold. Assume that $i, j, J_{i \cdot j} \in S_1^*$ and $i \boldsymbol{R}_1 j$. The *DC requires $i \boldsymbol{R}_1 J_{i \cdot j}$ and $J_{i \cdot j} \boldsymbol{R}_1 j$, which are not certain if either i or j is in C_1. Doing a similar analysis thoroughly leads to the next theorem.

Theorem 7. *Let f satisfy the API and SND properties, and let a composition satisfy the INV property. If a UIR satisfies for $J_{i \cdot j}, J_{j \cdot i} \in S_1 \cap S_2$,*

$$i, j \in S_1^* \ (S_2^*) \quad \textit{implies} \quad J_{i \cdot j} \in C_1 \ (C_2),$$

$$i \in S_1^*,\ j \in S_2^*,\ i \boldsymbol{R}_1 J_{i \cdot j} \ (J_{j \cdot i}) \textit{ and } j \boldsymbol{R}_2 J_{i \cdot j} \ (J_{j \cdot i}) \quad \textit{imply} \quad J_{i \cdot j} \ (J_{j \cdot i}) \in C_1,$$

$$i \in S_1^*,\ j \in S_2^*,\ J_{i \cdot j} \ (J_{j \cdot i}) \boldsymbol{R}_2 j \textit{ and } J_{i \cdot j} \ (J_{j \cdot i}) R_1 i \quad \textit{imply} \quad J_{i \cdot j} \ (J_{j \cdot i}) \in C_2,$$

and if the composition satisfies

$$i \boldsymbol{R}_u j \quad \textit{implies} \quad i \boldsymbol{R}_u J_{i \cdot j} \boldsymbol{R}_u j \textit{ and } i \boldsymbol{R}_u J_{j \cdot i} \boldsymbol{R}_u j \quad (u = 1, 2),$$

$$i \in S_1 \setminus S_2,\ j \in S_2 \setminus S_1 \textit{ and } J_{i \cdot j} \ (J_{j \cdot i}) \in S_1 \cap S_2 \quad \textit{imply} \quad i \boldsymbol{R}_1 J_{i \cdot j} \ (J_{j \cdot i}) \textit{ and/or } J_{i \cdot j} \ (J_{j \cdot i}) \boldsymbol{R}_2 j,$$

*then the composition satisfies the *DC.*

Proof. If $i \in S_1^*$, $j \in S_2^*$ and $J_{i \cdot j} \in S_1 \setminus S_2$, then i is in $S_1 \setminus S_2$ by the INV property. If $i \in S_1^*$, $j \in S_2^*$ and $J_{i \cdot j} \in S_2 \setminus S_1$, then j is in $S_2 \setminus S_1$. For these, the *DC is approved by Theorem 6 and III.3. If $i \in S_1^*$, $j \in S_2^*$ and $J_{i \cdot j} \in S_1 \cap S_2$, then either (1) $i \in S_1 \setminus S_2$, $j \in S_2 \setminus S_1$ or (2) $i \in C_1$, $j \in C_2$. For (1), the second line of the condition of the composition requires $i \boldsymbol{R}_1 J_{i \cdot j}$ and/or $J_{i \cdot j} \boldsymbol{R}_2 j$. If the former (latter), then the second (third) line of the condition of the UIR guarantees the *DC. If both hold, $J_{i \cdot j}$ can be in either of C_1 and C_2. For (2), III.2 requires $i \boldsymbol{R}_1 j$ and $j \boldsymbol{R}_2 i$, and/or $j \boldsymbol{R}_1 i$ and $i \boldsymbol{R}_2 j$. For the former (latter) the first condition of the composition gives $i \boldsymbol{R}_1 J_{i \cdot j}$ $(J_{i \cdot j} \boldsymbol{R}_2 j)$, and the second (third) line of the condition of the UIR guarantees the *DC. If $i, j \in S_1^*$ (S_2^*), let $J_{i \cdot j} \in S_1 \cap S_2$ because Theorem 6 guarantees the *DC for the other cases. The first condition of the UIR and the first condition of the composition approve the *DC. The proof for $J_{j \cdot i}$ is completed if $J_{i \cdot j}$ above is replaced by $J_{j \cdot i}$. □

The practical tie-breaking rule satisfies the condition in Theorem 7, if it is used with the following pre-flagging process after every composition: Assume temporarily $S_1^* = (S_1 \setminus S_2) \cup \{\text{jobs with flag 1 currently}\}$ and $S_2^* = (S_2 \setminus S_1) \cup \{\text{jobs with flag 2 currently}\}$; check the composed jobs and the new composite job for the lefthand side of the conditions of a UIR in Theorem 7, and if a condition fits for them, put a proper flag on the composite job. Implementing this process is very easy, because it suffices to check the three jobs. A UIR is assumed to satisfy the conditions in Theorem 7 from now on.

Theorem 8 (The $*$ *dominance boundedness* ($*$DB) of a composition). *Let f satisfy the API property and let a composition satisfy the $*$DC. Then,*

$$iR^*j \text{ and } iR^*k \quad \text{imply} \quad iR^*J_{j\cdot k} \text{ and } iR^*J_{k\cdot j}.$$

$$jR^*i \text{ and } kR^*i \quad \text{imply} \quad J_{j\cdot k}R^*i \text{ and } J_{k\cdot j}R^*i.$$

Proof. With no loss of generality, let jR^*k. By the $*$DC, $jR^*J_{j\cdot k}$, $jR^*J_{k\cdot j}$, $J_{j\cdot k}R^*k$ and $J_{k\cdot j}R^*k$. Moreover, the transitivity of R^* guarantees that $iR^*J_{j\cdot k}$ and $iR^*J_{k\cdot j}$ if iR^*j, and that $J_{j\cdot k}R^*i$ and $J_{k\cdot j}R^*i$ if kR^*i. □

Composite jobs defined on the flow-shop problem satisfy the properties introduced in this section except the SDC and the consistent equivalency. Use

$$f(a1\cdot a2)=f(J_{a1}\cdot J_{a2})+\sum_{i\in\{a1\}\cup\{a2\}}(A_i+B_i)-f(a1)-f(a2).$$

Then it is easy to see that

$$A_{a1\cdot a2}=f(a1\cdot a2)-\sum_{i\in\{a1\}\cup\{a2\}}B_i=f(J_{a1}\cdot J_{a2})-(B_{a1}+B_{a2})=A_{12}.$$

Similarly, $B_{a1\cdot a2}=B_{12}$; the coincidency. The INV and $*$IDN properties, and the conditions in Theorem 7 are shown to be satisfied immediately from the definition:

Table 1. Required properties and results

	Theorem 4 (R^*)	Theorem 7 ($*$DC)	Theorem 8 ($*$DB)	Theorem 9	Theorem 10	Corollary 13	Theorem 16
API	×	×	×	×	×	×	×
SND		×		×	×	×	
Existence of composition		×	×	×	×	×	×
Coincidency				×	×	×	
# INV		×					
# $*$IDN				×	×	×	
% $*$DC			×	×	×	×	
$*$Consistent monotonicity				either		×	
Consistent equivalency					×		
SDC					×		
Some cond. of composition and a UIR		×					

$*$: Properties concerning R^*.

#: Properties that are automatically satisfied if S_1 (or S_2) $=\emptyset$.

%: A property that is automatically satisfied if S_1 (or S_2) $=\emptyset$ and the SND property is satisfied.

$$f(i \cdot j) = A_i + B_j + \max\{A_j, B_i\},$$

$$A_{i \cdot j} = f(i \cdot j) - B_i - B_j, \quad \text{and} \quad B_{i \cdot j} = f(i \cdot j) - A_i - A_j.$$

The $*$ consistent monotonicity is also immediately shown for $\boldsymbol{R}^*$ defined previously for this problem. A class of problems equivalent to the one-machine minimum total weighted-completion time problem will be shown in Section 5 to satisfy the SDC and the consistent equivalency.

The first column of Table 1 summarizes the requirements to a sequencing problem. A property with $*$ in its name concerns a UIR. It is noteworthy that the INV and $*$IDN properties become evident if S_1 or S_2 is empty. Notice also that $\boldsymbol{R} = \boldsymbol{R}_u$ (i.e., the $*$DC is always satisfied when f satisfies the SND property) if S_v $(u \neq v)$ is empty.

4. A decomposition algorithm

The objective in this section is to reconstruct Sidney's [15] theory, by making use of the properties introduced in the preceding sections. The fundamentals of the theory are changed into a dominance relation under a UIR and composite jobs from an interval order and strings. The structure of the new theory is parallel with those of the previous one, and the logical structure of the proof of a theorem is roughly parallel to that of the corresponding theorem in the previous theory. Therefore, proofs of only the first two theorems are given. Notice that a reference to a corresponding proposition given to each theorem below does not imply that the applicability of the theorem is the same as that of the referent. Theorems here have wider applicability in the sense that various UIR's are permitted to be used in an algorithm though it is unique in the referent. This difference comes principally from the use of $\boldsymbol{R}^*$ instead of an interval order. The theory in this section is also more inclusive in the sense that it involves problems equivalent to the one-machine minimum total weighted-completion time problem excluded from the previous theory [15].

A pair (a, K), where K is a set of jobs and a is a schedule in $Q(K)$, is called a *p-optimal pair* of a constrained problem $P(N, G)$, if

(1) K is an initial set of G,

(2) a is an optimal schedule of $P(K, G)$,

(3) $J_a \boldsymbol{R}^* J_{a'}$, if a' is an optimal schedule of $P(M, G)$ where M is another initial set of G, and

(4) $J_a \in S_1^*$.

A pair (a, K) is called a *p*-optimal pair* of $P(N, G)$, if

(5) (a, K) is a p-optimal pair of G, and

(6) if (a', M) is another p-optimal pair of G, then M is not a proper subset of K.

A p-optimal pair is a pair of an initial set and its optimal schedule whose composite job is in S_1^* and is not dominated by the composite job associated with the

optimal schedule of any other initial set. A p^*-optimal pair is a p-optimal pair whose initial set is minimal. When pair (a, K) is p-optimal (p^*-optimal), a and K are also said to be p-optimal (p^*-optimal).

Theorem 9 [15, Lemma 3]. *In $P(N, G)$, let f satisfy the API and SND properties. Let a composition satisfy the coincidency, the $*$IDN property, the $*$DC and either the consistent equivalency or the $*$ consistent monotonicity. If (a, K) is p^*-optimal and $b(T)$ is an optimal schedule of $P(T, G)$ where T is a terminal set of G_K, then*

$$J_{b(T)}\boldsymbol{R}^*J_a \quad \text{and} \quad J_{b(T)} \in S_1^*.$$

Proof. If $T=K$, by virtue of the consistent equivalency or the $*$ consistent monotonicity of the composition, $J_a \equiv J_{b(T)}$ and the theorem is proved. Let T be a proper subset of K, and assume that the theorem does not apply to it, i.e.,

$$J_a R^* J_{b(T)}.$$

With no loss of generality, assume that

(A) T is a minimum proper subset of K to which the theorem does not apply. Define as follows with respect to $G_K: E_{K\setminus T} = \{(i,j) \in E : i, j \in K \setminus T\}$, $E_T = \{(i,j) \in E : i, j \in T\}$, $E'_K = E_{K\setminus T} \cup E_T$, and $G'_K = (K, E'_K)$. G'_K is obtained from G_K by removing all arcs between $K \setminus T$ and T. Because f satisfies the API and SND properties, $P(K, G'_K)$ has an optimal schedule of the following form:

$$a' = a1 \cdot b1 \cdot a2 \cdot b2 \cdot \cdots \cdot ar \cdot br \cdot a\,r+1,$$

where

(B) ai and bi are optimal schedules of $P(\{ai\}, G\}$ and $P(\{bi\}, G)$, respectively,

(C) $T = \bigcup_{i=1}^{r+1} ai$, $K \setminus T = \bigcup_{i=1}^{r} bi$,

(D) $J_{a1}\boldsymbol{R}^*J_{b1}\boldsymbol{R}^*J_{a2}\boldsymbol{R}^*J_{b2}\boldsymbol{R}^*\cdots\boldsymbol{R}^*J_{ar}\boldsymbol{R}^*J_{br}\boldsymbol{R}^*J_{a\,r+1}$,

and $a1$ and/or $a\,r+1$ may be empty. Note that $\{b1\}$ is an initial set of $G_{K\setminus T}$, and hence of G.

Let $r \geq 2$. $\{a\,r+1\}$ is a proper subset of T and a terminal set of G_K. By (A),

$$J_{a\,r+1}\boldsymbol{R}^*J_a.$$

On the other hand, because $\boldsymbol{R}^*$ is transitive,

$$J_{b1}\boldsymbol{R}^*J_a$$

holds by (D), and this implies that $b1$ is p-optimal (note that $b1$ is an initial set of G'_K). However, this is a contradiction to the fact that a is p^*-optimal. As a result, if $\{a\,r+1\}$ is not empty, then $r=1$. If $\{a\,r+1\}$ is empty, then substitute $a\,r$ for $a\,r+1$ in the above, and a contradiction is dervied for $r \geq 3$ and for $r=2$ and $\{a1\} \neq \emptyset$. If $r=2$ and $\{a1\} = \emptyset$, i.e., $a' = b1 \cdot a2 \cdot b2$, then a contradiction is derived by an inference similar to that for $r=1$ and $\{a1\} = \emptyset$ ($\{a2\} \neq \emptyset$) stated below. Therefore, even if $\{a\,r+1\}$ is empty, $r=1$.

Let $r=1$; then $a' = a1 \cdot b1 \cdot a2(\cdot\, b2)$. The last part ($\cdot\, b2$) is added for the sake of

convenience in order to express the case of $r=2$, $\{a1\}=\emptyset$ and $\{a3\}=\emptyset$. If $\{a1\}\neq\emptyset$ and $\{a2\}\neq\emptyset$, then a contradiction is derived by an inference similar to that for the case of $r=2$, $\{a1\}\neq\emptyset$ and $\{a3\}=\emptyset$. Therefore, at the first place, let $\{a1\}=\emptyset$. By the definition, $a2$ is an optimal schedule of $P(T,G)$. By virtue of the consistent equivalency or the $*$ consistent monotonicity, $J_{b(T)}\equiv J_{a2}$, and therefore, by the assumption, $J_a R^* J_{a2}$. If $\{b2\}\neq\emptyset$, then by the $*$DC of the composition, $J_{a2}\boldsymbol{R}^* J_{a2\cdot b2}$. Therefore, by the transitivity of $\boldsymbol{R}^*$, $J_a R^* J_{a2(\cdot b2)}$. Remember that a is p^*-optimal and $b1$ is not p-optimal, and it is evident that $J_a R^* J_{b1}$. Hence, the $*$DB guarantees

(E) $$J_a R^* J_{b1\cdot a2(\cdot b2)}.$$

If the composition satisfies the consistent equivalency, (E) is a contradiction because $\{a\}=\{b1\cdot a2(\cdot b2)\}=K$. Therefore, let the composition satisfy the $*$ consistent monotonicity. Because $J_{b1\cdot a2(\cdot b2)}\in S_1^*$ by the $*$IDN property, (E) implies $f(a)<f(b1\cdot a2(\cdot b2))$. This is a contradiction because $b1\cdot a2(\cdot b2)$ is an optimal schedule of $P(K,G'_K)$ which is associated with a relaxed precedence constraint of $P(K,G_K)$. As a result, $J_{a2}\boldsymbol{R}^* J_a$ holds, and this implies

$$J_{b(T)}\boldsymbol{R}^* J_a$$

because $J_{b(T)}\equiv J_{a2}$.

Next, let $\{a2\}=\emptyset$, i.e., $a'=a1\cdot b1$. By the assumption, $J_a R^* J_{a1}$. Moreover, by the p^*-optimality of a, $J_a R^* J_{b1}$. Hence, by the $*$DB, $J_a R^* J_{11}$, where J_{11} is the composite job for $J_{a1}\cdot J_{b1}$ and by the coincidency of the composition, $J_{11}\equiv J_{a1\cdot b1}$. Therefore, we conclude that $J_a R^* J_{a1\cdot b1}$. Following the inference below (E), it is shown that $J_{b(T)}\boldsymbol{R}^* J_a$.

Finally, by the definition of $\boldsymbol{R}^*$, $J_{b(T)}\in S_1^*$ because $J_a\in S_1^*$. □

Theorem 10 [13, Lemma 4]. *Let $P(N,G)$ and its associated composition satisfy the condition of Theorem 9. Moreover, let the composition satisfy the SDC property and the consistent equivalency. If (a,K) is a p^*-optimal pair of $P(N,G)$, then $a\cdot a(N\setminus K)$ is an optimal schedule of $P(N,G)$, where $a(N\setminus K)$ is an optimal schedule of $P(N\setminus K,G)$.*

Proof. Define $E_K=\{(i,j)\in E: i,j\in K\}$, $E_{N\setminus K}=\{(i,j)\in E: i,j\in N\setminus K\}$, $E'=E_K\cup E_{N\setminus K}$ and $G'=(N,E')$. $P(N,G')$ has an optimal schedule of the following form:

$$a1\cdot b1\cdot a2\cdot b2\cdot\cdots\cdot ar\cdot br\cdot a\,r+1.$$

By virtue of the SND and API properties of f, assuming that ai and bi are optimal schedules of $P(\{ai\},G)$ and $P(\{bi\},G)$, respectively, there is no loss of generality. Both or one of $a1$ and $a\,r+1$ may be empty, and

$$K=\bigcup_{i=1}^{r}\{bi\}\quad\text{and}\quad N\setminus K=\bigcup_{i=1}^{r+1}\{ai\}.$$

Let $r \geq 2$. $\{b1\}$ is an initial set of $P(N, G')$, and of $P(N, G)$. Assume that

$$J_{b1} R^* J_{b2} R^* \cdots R^* J_{br}.$$

By the $*$DC and the coincidency of the composition,

$$J_{b1} \boldsymbol{R}^* J_{b1 \cdot b2 \cdot \cdots \cdot br}.$$

Therefore, if $J_{b1} \in S_2^*$, then $J_{b1 \cdot b2 \cdot \cdots \cdot br} \in S_2^*$ by the definition of $\boldsymbol{R}^*$. However, this contradicts to the $*$IDN property, because $J_a \in S_1^*$. Therefore, $J_{b1} \in S_1^*$. Remember that $\{b1\} \subset \{a\}$, and $J_a R^* J_{br}$ by the transitivity of $\boldsymbol{R}^*$. However, this is a contradiction to Theorem 9 because $\{br\}$ is a terminal set of $P(K, G)$. As a result, $J_{b\,i+1} \boldsymbol{R}^* J_{bi}$ holds for some i in $\{1, 2, \ldots, r-1\}$. Then,

(i) if $J_{a\,i+1} R^* J_{b\,i+1}$, then $J_{a\,i+1} \boldsymbol{R}^* J_{bi}$ and $\cdots ai \cdot a\,i+1 \cdot bi \cdot b\,i+1 \cdot \cdots$ is another optimal schedule by the weak reasonability of $\boldsymbol{R}^*$, and

(ii) if $J_{b\,i+1} \boldsymbol{R}^* J_{a\,i+1}$, then $\cdots a_i \cdot bi \cdot b\,i+1 \cdot a\,i+1 \cdot \cdots$ is another optimal schedule by the weak reasonability of $\boldsymbol{R}^*$.

In either case of (i) and (ii), r is decreased by 1. Repeat this process until r is reduced to 1.

Let $r = 1$, then the optimal schedule of $P(N, G')$ is $a1 \cdot b1 \cdot a2$. By the SND property, $a1 \cdot a \cdot a2$ is another optimal schedule. We will show that $J_a \boldsymbol{R}^* J_{a1}$. In order to derive a contradiction, assume that $J_{a1} R^* J_a$. By the SDC, $J_{a1 \cdot a} R^* J_a$; besides, $J_{a1 \cdot a} \in S_1^*$ because $J_a \in S_1^*$. $K \cup \{a1\}$ is an initial set of G, and if c is an optimal schedule of $P(K \cup \{a1\}, G\}$, $J_c \in S_1^*$ by the $*$IDN property. Moreover, $J_{a1 \cdot a} \equiv J_c$ by the consistent equivalency, and this implies $J_c R^* J_a$; a contradiction to the p-optimality of a. As a result, $J_a \boldsymbol{R}^* J_{a1}$ is proved. Therefore, $a \cdot a1 \cdot a2$ is optimal, too. This schedule is feasible, and therefore, an optimal schedule of $P(N, G)$. By the SND property of f, $a \cdot a(N \setminus K)$ is also optimal. □

Theorem 10 asserts that $P(N, G)$ has an optimal schedule with a p^*-optimal subschedule as its first part, if the composition associated with it satisfies the SDC and the consistent equivalency. This gives a basis of a decomposition algorithm. A theorem similar to Theorem 10 can be established for $P(N, G)$ of which an associated composition satisfies the $*$ consistent monotonicity, by using a slightly different method.

A (sub-)schedule, s, of $P(N, G)$ determined by the S-algorithm below is called a *start sequence* of $P(N, G)$.

S-Algorithm

(1) $a \leftarrow \emptyset$, $M \leftarrow N$.

(2) Determine a p^*-optimal pair (a, K) of $P(M, G)$. If it does not exist, terminate.

(3) Terminate if $J_s \boldsymbol{R}^* J_a$. (Assume for the sake of convenience that $iR^* J_a$ for all $i \in S$ if $\{a\}$ is empty.)

(4) $s \leftarrow s \cdot a$, $M \leftarrow M \setminus K$.

(5) Terminate if $M = \emptyset$, otherwise go to (2).

In the S-algorithm, if there is no p^*-optimal pair at the first execution of the step (2), there is no start sequence (or, $s=\emptyset$). Then, $J_a \in S_2^*$ where a is an optimal schedule for an arbitrary initial set, and the reverse problem will apply to it.

Theorem 11 [15, Lemma 4]. *Let $P(N, G)$ and its associated composition satisfy the condition of Theorem 9. Moreover, let the composition satisfy the $*$ consistent monotonicity. Let a start sequence of $P(N, G)$ be $a = a1 \cdot a2 \cdot \cdots \cdot ah$, where $\{a\} = K$ and ai $(i = 1, 2, \ldots, h)$ is the i-th p^*-optimal sub-schedule determined at the step (2) of the S-algorithm. Then the following hold:*

(i) *For $i = 1, 2, \ldots, h$,*

$$J_{a1} \boldsymbol{R}^* J_{a1 \cdot a2 \cdot \cdots \cdot ai} \boldsymbol{R}^* J_{a1} \quad \text{and} \quad J_{a1 \cdot a2 \cdot \cdots \cdot ai} \in S_1^*.$$

(ii) *a is p-optimal schedule of $P(N, G)$.*

(iii) *For an optimal schedule $a(N \setminus K)$ of $P(N \setminus K, G)$, $J_a \boldsymbol{R}^* J_{a(N \setminus K)}$.*

(iv) *Let T and $a(T)$ be a terminal set of $P(K, G)$ and an optimal schedule of $P(T, G)$, respectively, then $J_{a(T)} R^* J_a$ and $J_{a(T)} \in S_1^*$.*

Theorem 12 below is proved by using Theorem 11. Note that Theorem 12 is different from Theorem 10 in the point that a is not a p^*-optimal sub-schedule but a start sequence. Notice also that the SDC and the consistent equivalency of a composition are replaced by the $*$ consistent monotonicity.

Theorem 12 [15, Theorem 5]. *Let $P(N, G)$ and its associated composition satisfy the condition of Theorem 11. Let a and $a(N \setminus K)$ be a start sequence of $P(N, G)$ such that $\{a\} = K$ and an optimal schedule of $P(N \setminus K, G)$, respectively. Then, $a \cdot a(N \setminus K)$ is an optimal schedule of $P(N, G)$.*

Remember that $a1$ in a is a p^*-optimal sub-schedule of $P(N, G)$, and Theorem 12 asserts that $P(N, G)$ has an optimal schedule of which the first part is a p^*-optimal schedule of $P(N, G)$. This proves Corollary 13 which parallels Theorem 10.

Corollary 13 [15, Corollary 6]. *Let $P(N, G)$ and its associated composition satisfy the condition of Theorem 11. Let (a, K) and $a(N \setminus K)$ be a p^*-optimal pair of $P(N, G)$ and an optimal schedule of $P(N \setminus K, G)$, respectively. Then $a \cdot a(N \setminus K)$ is an optimal schedule of $P(N, G)$.*

The Core Algorithm

(1) $a \leftarrow \emptyset$, $M \leftarrow N$, $H \leftarrow G$.

(2) Repeat 1 and 2 below as long as $P(M, H)$ has a p-optimal initial set, then go to (3).

1. Determine a p^*-optimal pair $(a(K), K)$ of $P(M, H)$.
2. $a \leftarrow a \cdot a(K)$, $M \leftarrow M \setminus K$.

(3) Terminate.

Theorem 14. *Let $P(N, G)$ and its associated composition satisfy the condition either of Theorem* 10 *or of Theorem* 11. *Let a be the schedule determined by the core algorithm. Then $P(N, G)$ has an optimal schedule where a constitutes its first part.*

Theorem 14 is proved by repeatedly applying Theorem 10 or Corollary 13 to $P(N, G)$. If $M=\emptyset$ at the step (3) of the core algorithm, then a alone is an optimal schedule of $P(N, G)$ by Theorem 14. if M is not empty, the composite job for an optimal schedule of every initial set of the current $P(M, H)$ is an element of S_2^*. An optimal schedule of this $P(M, H)$ can be determined by applying the core algorithm to its reverse problem.

Define $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{R}}_u$ $(u=1,2)$ for $\tilde{P}(N, \tilde{G})$ by

$$i\tilde{\boldsymbol{R}}j \text{ iff } j\boldsymbol{R}i, \text{ and } i\tilde{\boldsymbol{R}}_u j \text{ iff } j\boldsymbol{R}_u i.$$

Let $\tilde{S}_1=S_2$, $\tilde{S}_2=S_1$, $\tilde{C}_1=C_2$ and $\tilde{C}_2=C_1$.

It is easy to show that if the original problem satisfies a property described in the preceding sections, then its reverse problem satisfies the same property. It works well to formally transform inferences required for the reverse problem into ones for the original problem. Therefore, Theorem 15 is easily proved.

A Decomposition Algorithm

(1) Perform the core algorithm on $P(N, G)$, and let $P(M, H)$ and a_1 be the remaining problem and the obtained schedule, respectively.

(2) Perform the core algorithm on $\tilde{P}(M, \tilde{H})$, and, let $\tilde{a}_2$ be the obtained schedule.

(3) $a \leftarrow a_1 \cdot a_2$, then terminate.

Theorem 15 [15, Theorem 8]. *Let $P(N, G)$ and its associated composition satisfy the condition either of Theorem* 10 *or of Theorem* 11. *Schedule a obtained by the decomposition algorithm is an optimal schedule of $P(N, G)$.*

In the following, the problem instance in Fig. 1 of the flow-shop problem is solved by the decomposition algorithm.

When step (1) of the decomposition algorithm is first performed, initial sets which can be p^*-optimal at (2) 1 of the core algorithm are $K_1=\{1,2\}$, $K_2=\{1,3\}$, $K_3=\{1,2,3\}$ and $K_4=\{1,2,3,5\}$. Composite jobs for optimal schedules of these initial sets are determined as follows. By the definition in Section 2, $T_1(1)=A_1=3$, $T_2(1)=\max\{T_1(1),0\}+B_1=5$, $T_1(1\cdot 2)=T_1(1)+A_2=5$, $T_2(1\cdot 2)=\max\{T_1(1\cdot 2), T_2(1)\}+B_2=9=f(1\cdot 2)$. Therefore, $A_{1\cdot 2}=f(1\cdot 2)-(B_1+B_2)=3$, $B_{1\cdot 2}=f(1\cdot 2)-(A_1+A_2)=4$, i.e., $J_{1\cdot 2}=(A_{1\cdot 2}, B_{1\cdot 2})=(3,4)$. Similarly, $J_{1\cdot 3}=(3,4)$. $J_{1\cdot 2\cdot 3}=J_{1\cdot 3\cdot 2}=(3,6)$ and $J_{1\cdot 2\cdot 3\cdot 5}=J_{1\cdot 3\cdot 2\cdot 5}=(3,5)$. Therefore, the first pair can be either $(1\cdot 2, \{1,2\})$ or $(1\cdot 3, \{1,3\})$. If the former is adopted, the second p^*-optimal pair is $(3, \{3\})$, and if the latter is adopted, the second is $(2, \{2\})$. Anyway, the third is $(5, \{5\})$, and then (1) of the decomposition algorithm terminates. At that time,

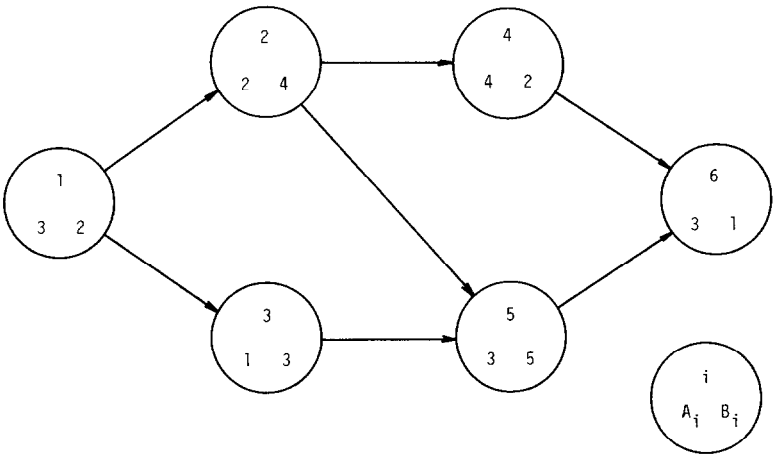

Fig. 1. A problem instance of the two-machine minimum makespan flow-shop problem.

$a_1 = 1 \cdot 2 \cdot 3 \cdot 5$ or $a_1 = 1 \cdot 3 \cdot 2 \cdot 5$, $M = \{4, 6\}$ and $\tilde{E}_M = \{(6, 4)\}$. Perform (2) of the decomposition algorithm on $\tilde{P}(M, \tilde{H})$ where $\tilde{H} = (M, \tilde{E}_M)$. Then the first and the second p^*-optimal pairs are $(6, \{6\})$ and $(4, \{4\})$, respectively, and $\tilde{a}_2 = 6 \cdot 4$. Finally, $a = 1 \cdot 2 \cdot 3 \cdot 5 \cdot 4 \cdot 6$ or $a = 1 \cdot 3 \cdot 2 \cdot 5 \cdot 4 \cdot 6$ is obtained as an optimal schedule of $P(N, G)$.

The decomposition algorithm is not necessarily very efficient, because no efficient algorithm finding a p^*-optimal pair has been proposed yet. A problem instance may not be decomposed at all by the decomposition algorithm. Actually, most sequencing problems under a general precedence network are NP-complete [2]. However, Theorem 16 shows that a problem under a general precedence constraint can be reduced with respect to a sub-schedule corresponding to an arc which is a chain.

Theorem 16 [12, Theorem 1], [8, Theorem 2]. *Assume that f of $P(N, G)$ satisfies the API property and a composition is defined. If $j\mathbf{R}i$ for an arc (i, j) which is a chain of G, then $P(N, G)$ has an optimal schedule in which i directly precedes j.*

5. Applications

Three classes of problems illustrate applicability of the proposed theory. Example 1 shows a class of problems for which Theorem 10 is applicable. This class was excluded from the previous theory [15], though a specific decomposition algorithm had been given [13]. Example 2 shows a class of problems for which Theorem 12 (Corollary 13) is applicable. This class is the one included in the previous theory. Problems in these two classes can be solved by the decomposition algorithm (Theorem 15). The third class of problems in Example 3 are not solved by the decomposition algorithm.

Finally, in Example 4, two numerical instances of the two-machine minimum makespan flow-shop problem show advantages of a UIR over the tie-breaking rule used in Sidney's interval order.

Example 1. The minimum total expected cost fault detection problem – a case of disjoint faults.

Suppose that in a system with n components needs to be found a faulty component. Component i may have a fault, and its probability is p_i. The faults of components are disjoint, i.e., $\sum_{i\in N} p_i = 1$, where N is the set of components (or their labels). The cost for inspecting whether component i is faulty or not is c_i, and the probability that an inspection of component i may overlook a fault is q_i. When $q_i \neq 0$, component i may be inspected more than one time. Inspections are continued until a fault is found (or, if a sequence of inspections is finite, it may be exhausted before a fault is found). The probability that the k-th inspection on component i finds a fault first is given by

$$X_{i,k} = p_i q_i^{k-1}(1-q_i).$$

Then, the set of components which *possibly* become components of a system of this type is denoted as

$$\hat{S} = \{(c_i, p_i, q_i) : 0 < p_i \leq 1,\ 0 \leq q_i < 1,\ c_i \text{ is a real}\}.$$

A specific system is described by

$$\hat{N} = \left\{(c_i, p_i, q_i) \in \hat{S} : i = 1, 2, \ldots, n, \sum_{i=1}^{n} p_i = 1\right\}.$$

The problem to be solved is to find a sequence of inspections which minimizes the total expected cost.

In order to explicitly describe the elements to be ordered, define

$$S = \{(c_i, X_{i,k}) : (c_i, p_i, q_i) \in \hat{S},\ k = 1, 2, \ldots\},$$

$$N = \{(c_i, X_{i,k}) : (c_i, p_i, q_i) \in \hat{N},\ k = 1, 2, \ldots\}.$$

Then, $Q(N)$ of the problem is the set of infinite sequences where the k-th inspection on component i precedes the $k+1$-th inspection on the same component. Then, the total expected cost of $a \in Q(N)$ is given [1] by

$$f(a) = \sum_{i=1}^{\infty} \left(\sum_{j=1}^{i} Y_{\langle j\rangle}\right) Z_{\langle i\rangle},$$

where $Y_{\langle j\rangle}$ is the cost of the j-th inspection in a (i.e., one of c_i's), and $Z_{\langle i\rangle}$ is the probability that a fault is first found by the i-th inspection in a (i.e., one of $X_{i,k}$'s).

The sequencing function can be equivalently transformed into

$$f(a) = \sum_{j=1}^{\infty} \left(\sum_{i=j}^{\infty} Z_{\langle i\rangle}\right) Y_{\langle j\rangle}.$$

Using this expression, the following dominance relation is easily induced. Denote the k-th inspection of component r as (r, k). Define $\boldsymbol{R}_1$ as

$$(r,k)\boldsymbol{R}_1(s,u) \quad \text{iff} \quad c_r/X_{r,k} \leq c_s/X_{s,u},$$

and let $S_1 = S$ and $S_2 = \emptyset$. Note that $\boldsymbol{R}_1$ gives only schedules in $Q(N)$ because $c_r/X_{r,k} \leq c_r/X_{r,k+1}$. It is easy to show that f of this problem satisfies the SND property as well.

Define the composite job $J_a = (c_a, X_a)$ for schedule a as follows:

$$c_a = \sum_{(r,k)\in\{a\}} c_r, \qquad X_a = \sum_{(r,k)\in\{a\}} X_{r,k} \quad \text{and} \quad \Delta_a = f(a) - X_a c_a.$$

This composition satisfies the consistent equivalency, the coincidency and the SDC. Notice that the INV and *IDN properties and the *DC are satisfied also.

If $q_i = 0$ for some i, define S and N such that they include one element with $k = 1$ in relation to such i. Particularly, if $q_i = 0$ for all i, N has only n elements. However, the sequencing functions for such cases have no serious difference from the preceding one, an the results do not change.

In the preceding analysis, we use the fact that c_r is a real and $X_{r,k}$ is a positive real, and do not use the fact that $X_{r,k}$ represents a probability. It is easy to see that the preceding analysis with $q_i = 0$ for all i applies to the one-machine minimum total weighted-completion time problem [16, 13] where job i is associated with a processing time c_i and a weight p_i on its completion time. The one-machine minimum average-completion time problem [16] is a special case of this problem.

Example 2. The minimum total expected cost fault detection problem – a case of independent faults.

Symbols are used with the same meaning as those of Example 1. The difference of the problems here and Example 1 is that the faults are assumed to be independent, rather than disjoint (i.e., the condition $\sum p_i = 1$ of $\hat{N}$ is omitted). The probability that the i-th inspection in schedule a is executed depends on which components are inspected and how many times the inspection on each component is executed during the 1st and i-1 st inspections. For the convenience of description, let

$$\prod_{j=u}^{u-1} (1 - Z_{\langle j\rangle}) = 1.$$

Because the probability that the j-th inspection does not find a fault is $1 - Z_{\langle j\rangle}$, the total epected cost to be minimized is given by

$$f(a) = \sum_{i=1}^{\infty} \left\{ \prod_{j=1}^{i-1} (1 - Z_{\langle j\rangle}) \right\} Y_{\langle i\rangle}.$$

It is easy to see that f of this problem satisfies the API and SND properties and that the dominance relation is defined as the same as that in Example 1.

Define the composite job $J_a=(c_a, X_a)$ for schedule a as follows:

$$c_a=f(a), \qquad X_a=1-\prod_{(r,k)\in\{a\}}(1-X_{r,k}) \quad \text{and} \quad \Delta_a=0.$$

Then it is easily shown to satisfy the coincidency, the SDC and the $*$ consistent monotonicity. As in Example 1, the results do not change even if $q_i=0$ for some or all i.

According to Kelly [3], Example 1 and the minimum maximum-cumulative excess cost problem [8] are special cases of Example 2 in the following sense: Let f' be the sequencing function of Example 1 or of the minimum maximum-cumulative excess cost problem. Then it is possible to define an instance of Example 2 corresponding to each instance of these problems such that for any pair of feasible schedules, a and b, of these problems,

$$f'(a)\le f'(b) \quad \text{implies} \quad f(a)\le f(b).$$

It is easy to show that the minimum maximum-cumulative excess cost problem is a generalization of both the flow-shop problem used in the preceding sections [8] and the one-machine minimum maximum-lateness problem [16]. According to Monma and Sidney [8], the one-machine minimum total discounted cost (with respect to completion time) problem is equivalent to Example 2.

Example 3. The minimum maximum-cost fault detection problem.

Symbols are used with the same meaning as those in Examples 1 and 2. Consider the case of independent faults. Moreover, let $q_i=0$ for all i. Then,

$$S=\{(c_i, p_i):(c_i, p_i, 0)\in\hat{S}\}, \quad \text{and} \quad N=\{(c_i, p_i)\in S: i=1,2,\ldots,n\}.$$

The sequencing function to be minimized is

$$f(a)=\max_{1\le i\le n}\left\{\prod_{u=1}^{i-1}(1-p_{\langle u\rangle})\right\}c_{\langle i\rangle}.$$

The dominance relation of this problem is defined as

$$\begin{aligned} iRj \quad \text{iff} \quad & c_i\le 0,\ c_j\le 0 \text{ and } c_i/(1-p_i)\ge c_j/(1-p_j), \text{ or} \\ & c_i\le 0 \text{ and } c_j\ge 0, \text{ or} \\ & c_i\ge 0,\ c_j\ge 0 \text{ and } c_i\le c_j. \end{aligned}$$

Define

$$\begin{aligned} & S_1=\{(c_i, p_i)\in S: c_i\le 0\}, \\ & S_2=\{(c_i, p_i)\in S: c_i\ge 0\}, \\ & iR_1j \quad \text{iff} \quad c_i/(1-p_i)\ge c_j/(1-p_j) \quad \text{for } i,j\in S_1, \\ & iR_2j \quad \text{iff} \quad c_i\le c_j \quad \text{for } i,j\in S_2. \end{aligned}$$

It is easy to see that f satisfies the API and SND properties. Define a job $J_a=(c_a, p_a)$ for schedule a by

$$c_a=f(a), \qquad p_a=1-\prod_{i\in\{a\}}(1-p_i) \quad \text{and} \quad \Delta_a=0.$$

This job satisfies the definition of composite jobs. However, it satisfies neither the consistent equivalency nor the $*$ consistent monotonicity. If faults are disjoint rather than independent, it is possible to show the API property of the sequencing function. However, defining composite jobs for this case is not possible.

Example 4. Advantage of flexibility of a UIR.

Consider the problem in Fig. 2(a). The start of processing job 3 is better to be

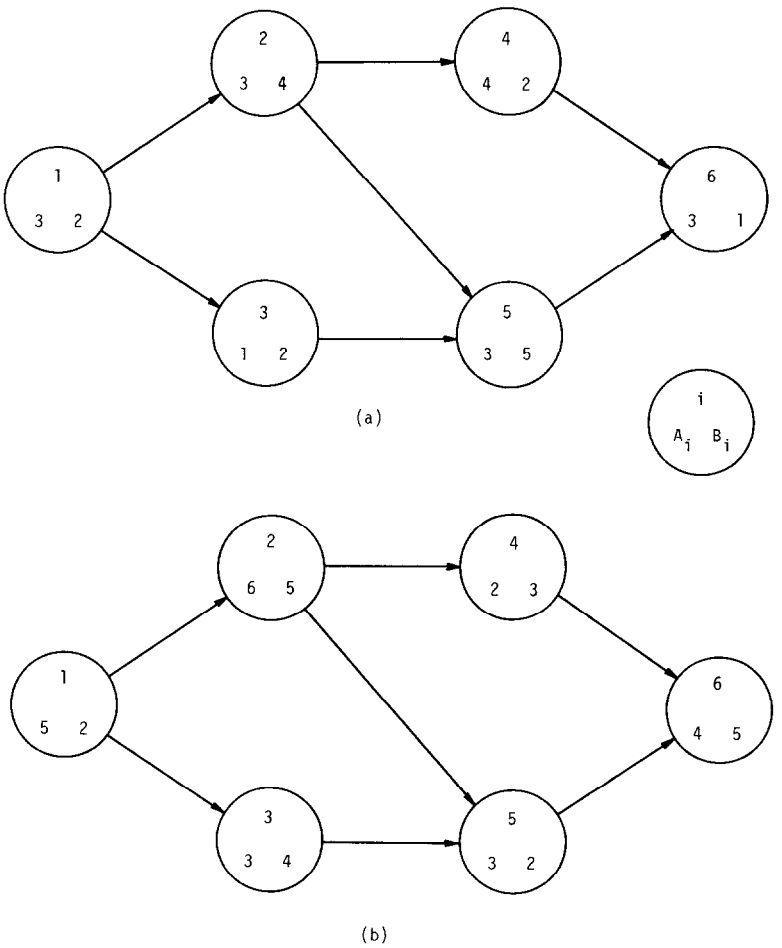

Fig. 2. Problem instances of the two-machine minimum makespan flow-shop problem showing advantages of a UIR.

put off as long as the shop does not become idle (because, for example, the material for job 3 is often delivered behind the scheduled date). Or, conversely, job 2 is better to be started processing as soon as possible because of some reason. $J_{1\cdot 2}=(4,4)$ and $J_{1\cdot 3}=(3,3)$. Assume that only a tie-breaking rule where $S_1^*=S_1$ and $S_2^*=S_2\setminus S_1$ is permitted (as in [15]). $J_{1\cdot 3}R^*J_{1\cdot 2}$ and the first p^*-optimal pair is $(1\cdot 3,\{1,3\})$. Therefore, job 3 is inevitably scheduled before job 2. Notice that a UIR in the proposed algorithm permits a tie-breaking rule to classify $J_{1\cdot 2}$ in S_1^* and $J_{1\cdot 3}$ in S_2^*, respectively, and job 2 can be scheduled before job 3.

Consider the problem in Fig. 2(b). Check optimal schedules of each initial set for its composite job, and all composite jobs are in $S_2\setminus S_1$, i.e., no p^*-optimal pair exists. Therefore, its reverse problem will be examined. Assume that only a tie-breaking rule with $S_1^*=S_1$ and $S_2^*=S_2\setminus S_1$ is allowed. Then, there is no initial set of which an optimal schedule gives a composite job in $\tilde{S}_1^*=S_2\setminus S_1$. Therefore, the problem is not decomposed at all by such a decomposition algorithm (i.e., such a decomposition algorithm becomes ineffective). Notice that a UIR in the proposed algorithm permits a tie-breaking rule to classify $J_{6\cdot 5}$ in $\tilde{S}_1^*$ and $J_{4\cdot 2}$ in $\tilde{S}_1^*$, i.e., it decomposes the problem.

6. Summary

Based on a dominance relation with a UIR and composite jobs, is proposed a revised theory of a decomposition algorithm. Required properties of sequencing problems and major results are related in Table 1. The proposed theory permits to use various tie-breaking rules, though a unique tie-breaking rule is used in the previous theory. Besides, the new theory includes a class of problems that are not included in the previous theory. In spite of these merits, it does not permit to use a tie-breaking rule in $\boldsymbol{R}'$ of Section 3. Apply the method in this paper to the series-parallel algorithm in [8], and a similar improvement emerges. Remember that III.3 of the properties of a dominance relation is used in the proofs of (4) in Theorem 1 and Theorem 2 and Theorem 7 only. These theorems are not necessary in the theory of a series-parallel algorithm, i.e., III.3 may be omitted from the definition of the API property for a series-parallel algorithm.

Acknowledgment

The authors gratefully acknowledges an anonymous referee for his or her helpful comments. This research was partly done at University of Pennsylvania under Fulbright Grant #07242.

References

[1] M.H. DeGroot, Optimal Statistical Decisions (McGraw Hill, New York, 1970) Chapter 14.14–14.15.

[2] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation on deterministic sequencing and scheduling, Mathematical Centre Tracts 99 (1978) 169–231.

[3] F.P. Kelly, A remark on search and sequencing problems, Math. Oper. Res. 7 (1982) 154–157.

[4] T. Kurisu, Two-machine scheduling under required precedence among jobs, J. Oper. Res. Soc. Japan 19 (1976) 1–13.

[5] E.L. Lawler, Sequencing jobs to minimize total weighted completion time subject to precedence constraints, in: B. Alspach, P. Hell and D.J. Miller eds., Annals Discrete Math. 2 (North-Holland, Amsterdam, 1978) 75–90.

[6] P.L. Maggu, G. Das and R. Kumar, On equivalent job for job-block in $2 \times n$ sequencing problems with transportation times, J. Oper. Res. Soc. Japan 24 (1981) 136–146.

[7] G.B. McMahon and P.G. Burton, Flow shop scheduling with the branch-and-bound method, Oper. Res. 15 (1967) 473–481.

[8] C.L. Monma and J.B. Sidney, Sequencing with series-parallel precedence constraints, Math. Oper. Res. 4 (1979) 215–224.

[9] C.L. Monma, Two-machine flow-shop problem with series-parallel precedence relations: an algorithm and extensions, Oper. Res. 27 (1979) 792–798.

[10] Y. Sekiguchi, Inter- and intra-group-of-jobs schedule for minimizing makespan in a two-machine GT shop, in: Preprints of IFAC 8th Triennial World Congress, Kyoto, Japan (1981) XIV164–XIV169.

[11] Y. Sekiguchi, Optimal schedule in a GT-type flow-shop under series-parallel precedence constraints, J. Oper. Res. Soc. Japan 26 (1983) 226–252.

[12] Y. Sekiguchi, Optimal schedules under series-parallel precedence constraints, Hokudai Economic Papers 13 (1984) 100–115.

[13] J.B. Sidney, Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs, Oper. Res. 23 (1975) 283–298.

[14] J.B. Sidney, The two-machine flow time problem with series-parallel precedence constraints, Oper. Res. 27 (1979) 782–791.

[15] J. Sidney, A decomposition algorithm for sequencing with general precedence constraints, Math. Oper. Res. 6 (1981) 190–204.

[16] W.E. Smith, Various optimizers for single-stage production, Naval Res. Logist. Quart. 3 (1956) 59–66.